


Webアプリケーション入門 🚀

WebアプリケーションとHTML

はじめに：今日やること

- 今日のゴール: 
 - Webアプリケーションがどういうものかを知る
 - HTMLの基本的な書き方を知る

Webアプリケーションってなに？

- スマホやパソコンの**ブラウザ**（Chrome, Safariなど）で使うサービスやソフトウェアのことだよ！
- **インストール不要**！特定のURLを開けばすぐに使える！
- WindowsでもMacでもスマホでも、**どの端末でも大体同じように使える**！

身近なWebアプリケーション

実は普段からたくさん使ってる！

- Webメール (Gmail, Outlook.com)
- 動画サイト (YouTube)
- SNS (X, Instagram, Facebook)
- ネットショッピング (Amazon, 楽天市場)
- オンラインオフィスツール (Google Docs, Sheets)
- 各種予約サイト (ホテル、飛行機)

などなど…

どうやって動いてるの？（全体像）

Webアプリは大きく分けて2つの仕組みで動いてるよ！

 見た目を作る仕組み（クライアントサイド）

↓ ↑

（インターネット上で会話してる！HTTP/HTTPS）

↓ ↑

 裏で仕事をする仕組み（サーバーサイド）

見た目を作る仕組み（クライアントサイド）

ブラウザ上で、みんなが見る画面（表側）を作ってるよ！

- **HTML:** ページの**骨組み**を作る（設計図）
 - 文字、画像、リンクなどを配置
- **CSS:** 見た目を**キレイに飾る**（インテリア）
 - 色、大きさ、配置、アニメーションなどを調整
- **JavaScript:** ページに**動きをつける**（仕掛け）
 - ボタン操作、入力チェック、サーバーとの通信など

⇒ これらを解釈して画面に表示するのが「**Webブラウザ**」

裏で仕事をする仕組み（サーバーサイド）

インターネットの向こう側で、みんなには見えない処理（裏側）を担当！

- **Webサーバー**: ブラウザからの「お願い」を受け取る**窓口**
- **APサーバー & プログラム**: アプリの**頭脳**！実際の処理を担当 (Ruby, Python, PHPなどで書かれる)
- **データベース**: 会員情報や商品リストなど、大事なデータをしまっておく**倉庫**
- **インフラ**: 全体が動くための**土台**（サーバーPC、ネットワーク、クラウドなど）

会話のルール (HTTP/HTTPS)

クライアントとサーバーが会話（通信）するときのルールだよ！

- 基本ルール → **HTTP** (HyperText Transfer Protocol)
- **安全**な会話（暗号化） → **HTTPS** (HTTP Secure)
 - 今はほとんどこっち！アドレスバーに🔒マーク！

会話の流れ

1. ブラウザ：「お願い！」（リクエスト）
2. サーバー：「はい、どうぞ！」（レスポンス）

お願いの種類①：GET (ちょうだい！)

サーバーから情報をもらいたい時のお願い方法。

- **目的:** ページやデータなどを**取得する (Get)**
- **例:** Webページ表示、検索結果表示
- **特徴:**
 - 欲しい情報は**URLにくっつけて送る** (`?q=キーワード`)
 - → URLに情報が見えちゃう👁👁
 - → **秘密の情報** (パスワードとか) は送っちゃダメ！

お願いの種類②：POST (これお願い！)

サーバーにデータを送って処理してほしい時のお願い方法。

- **目的:** データを送信して (Post)、登録・更新などしてもらう
- **例:** ログイン、会員登録、フォーム送信、SNS投稿
- **特徴:**
 - データは**見えない小包 (ボディ) **で送る
 - → URLには見えない (GETより安全)
 - → **秘密の情報**も送れる

GETとPOSTの使い分けまとめ

やりたいこと	おすすめ	特徴
情報を見る・もらう (検索など)	GET	URLで見える, ブックマーク可, 安全性は低い
データを送る・登録する (ログイン等)	POST	URLで見えない, ブックマーク不可, GETより安全

基本は…

- サーバーの情報を**変えない** → GET
- サーバーの情報を**変える可能性がある** → POST

見た目を作る仕組み

HTMLの基本

HTMLって、そもそも何？ (1/2)

- ウェブページの「見た目の骨組み」を作る特別な「言葉」
- 正式名称: HyperText Markup Language
 - (ハイパーテキスト マークアップ ランゲージ)
 - 難しく考えなくてOK！
- 例：文字や画像を「どこに」「どんな順番で」表示するか決める

HTMLって、そもそも何？ (2/2)

家づくりで例えると？

- **HTML = 設計図**
 - (柱、壁、窓の位置を決める)
- **ブラウザ (Chrome, Safariなど) = 大工さん**
 - (設計図を読んで家を建てる！)

HTMLで「指示書」を作り、ブラウザがそれを読んで画面に表示します。

HTMLの基本ルール - 全体の構成

HTMLファイルはだいたいこんな形になっています。

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <title>ページのタイトル</title>
  </head>
  <body>
    ここに内容を書くよ！
  </body>
</html>
```

順番に見ていきましょう！

ルール: 要素とタグ

- **要素:** ページを構成する「部品」
 - 例: 段落、見出し、画像、リンクなど
- **タグ:** 部品の種類を示す「ラベル」(< > で囲む)
 - **開始タグ:** <p> (ここから部品が始まるよ)
 - **終了タグ:** </p> (ここで部品が終わるよ / スラッシュ付き)
- ほとんどの要素は開始タグと終了タグで内容を挟みます。

ルール: DOCTYPE宣言

```
<!DOCTYPE html>
```

- 「このファイルはHTML5ルールで書かれています！」という宣言。
- HTMLファイルの一番最初に必ず書くおまじない。

ルール: `<html>` タグ

```
<html lang="ja">  
</html>
```

- ページ全体の「おおもと」。
- すべての要素（部品）はこの `<html>` タグの中に入れます。
- `lang="ja"` は「このページは日本語ですよ」という印。

ルール: `<head>` タグ

```
<head>  
  <meta charset="UTF-8"> <title>ブラウザのタブに出るタイトル</title>  
</head>
```

- ページの「裏方情報」置き場。
 - （画面には直接表示されない情報）
- 例：ページのタイトル、文字コード、CSSファイルの場所など。

ルール: `<body>` タグ

```
<body>
  <h1>大きな見出し</h1>
  <p>ここに文章を書きます。</p>
  
</body>
```

- ページの「見える」部分。
- ここに書いたものがブラウザの画面に表示されます！
- テキスト、見出し、画像、リンクなど、表示したい内容は全部ここ。

ルール: 要素とタグ (続き)

空要素 (Empty Element)

- 中には、内容を持たず**終了タグがない**特別な部品もあります。
- 開始タグだけで完結します。
- 例:
 - `` : 画像を表示
 - `
` : 改行
 - `<hr>` : 水平線
 - `<input>` : フォームの入力欄

ルール: 属性 (Attribute)

- タグに「追加情報」や「詳細設定」を与えるもの。
- 開始タグの中に `属性名="値"` の形で書き加えます。
- 例1: リンク先 (`href`) を指定

```
<a href="[https://google.com](https://google.com)">Googleへのリンク</a>
```

- 例2: 画像の場所 (`src`) と代替テキスト (`alt`) を指定

```

```

ルール: 入れ子 (Nesting)

- 要素（部品）の中に、さらに別の要素を入れること。
 - （マトリョーシカ人形みたい！）
- **正しい順番**で開始し、**入れたのと逆順**で閉じるのがルール。
- **○ OK例:**

```
<p>これは<strong>重要な</strong>ポイントです。</p>
```

- **✗ NG例:** (閉じ方が間違い)

```
<p>これは<strong>重要</p></strong>
```

ルール: コメント

```
<!-- これはコメントです。ブラウザには表示されません。 -->
```

- `<!-- -->` で囲んだ部分は「メモ」。
- ブラウザには表示されません。
- コードを分かりやすくするために使います。

まとめ

- Webアプリケーションは、クライアントとサーバーの仕組みで動いている。
- 会話のルールはHTTP
- クライアント側はHTML, CSS, JavaScriptで見た目を作る。
- HTMLはページの骨組みを作るための言語。
- HTMLの基本ルールを理解して、正しく書けるようになるう！